Recent Progress in Nonlinear and Linear Solvers

P A Lott[1], H C Elman[2], K J Evans[3], X S Li[4], A G Salinger[5], C S Woodward[1]

[1] Lawrence Livermore National Laboratory, Livermore, CA 94550, USA
[2] University of Maryland, College Park, MD 20742, USA
[3] Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA
[4] Lawrence Berkely National Laboratory, Berkeley, CA 94720, USA
[5] Sandia National Laboratories, Albuquerque, NM 87185, USA

E-mail: Aaron.Lott@llnl.gov

Abstract. We discuss two approaches for tackling algebraic systems; one is based on block preconditioning, and the other is based on multifrontal and hierarchical matrix methods. First we consider a new preconditioner framework for supporting implicit time integration within an atmospheric climate model. We give an overview of the computational infrastructure used in atmospheric climate studies, address specific challenges of weak scalability of numerical methods used in these codes, outline a strategy for addressing these challenges, and provide details about the software infrastructure being developed to implement these ideas. In the second part, we present our recent results of employing hierarchically semiseparable low-rank structure in a multifrontal factorization framework. This work work leads to superfast linear solvers for elliptic PDEs and effective preconditioners for a wider class of sparse linear systems.

1. Enabling Scalable Solvers in Atmospheric Climate Models

Understanding the sensitivity of Earth's climate to radiative forcing requires ensemble forecasts over long time periods. In order to resolve local phenomena, such as hurricanes, and measure variability at the decadal scale, high resolution global simulations are needed. Improvements in numerical methods and parallel computing architectures have led to significant modeling capability [1]. However, long-term high-resolution climate studies require further algorithmic advances in order to fully utilize these and future architectures.

The Community Earth System Model (CESM) provides coupled atmospheric, land, ocean, and ice climate models to address scientific questions about the Earth's climate and to provide information for making policy decisions. The Community Atmospheric Model (CAM) is the atmospheric model component of CESM. Atmospheric dynamics are integrated forward in time through a dynamical core. A spectral element dynamical core (CAM-SE) based on the High-Order Method Modeling Environment (HOMME) has strong scaling features, making it a favorable choice for high-resolution simulation. HOMME uses a cubed sphere, spectral-element discretization of the Earth and supports two mathematical models for the atmosphere, the shallow water equations and the primitive equations. In this paper, we discuss numerical methods being developed to support the shallow water component in HOMME [2].

The numerical methods used in the default branch of HOMME are explicit. Numerical stability requirements of these methods force the maximum time step to be less than the minimum grid size divided by the maximum flow velocity, the CFL condition [3]. This condition causes time steps to become small when simulating atmospheric flows at high resolution. For a spectral-element method

of order N, with M elements, the grid points within each element are spaced like $1/N^2$ near element boundaries; this causes the time step Δt to be inversely proportional to MN^2. The superlinear computational complexity in spatial resolution expressed through the CFL constraint prevents weak scaling of explicit methods [4].

Implicit methods allow time steps to be taken independent of grid resolution and can provide a weak scalable alternative to explicit methods. Implicit methods have recently been added to HOMME by interfacing with nonlinear and linear solvers provided through Trilinos [5]. At each timestep a matrix-free Newton's method is used as the nonlinear solver. In each nonlinear iteration, a Krylov iterative method solves the subsidiary linear system. The Jacobian-vector product needed by the Krylov iteration is approximated by a finite difference directional derivative of the residual vector. This implicit code matches the strong scaling of the explicit code when a similar size time step is used [6]. Additionally with the implicit method, timesteps can be taken 30–60 times greater than the gravity wave stability limit and still achieve results where the L_2 error is of the same order as the explicit scheme [7]. The caveat of this implicit approach, however, is the cost of the linear system solve, which creates a bottleneck in the simulation.

The linear system to be solved at each nonlinear iteration is sparse and nonsymmetric with dense subblocks corresponding to the operator defined within each element. Iterative methods can be used to solve this system, but the rate of convergence of these methods depends on matrix properties such as the condition number and spread of eigenvalues. Preconditioners, which can be viewed as matrices that transform a linear system into a system with improved properties, are used to accelerate the convergence of iterative methods. The effectiveness of a preconditioner relies on its ability to mimic the spectrum of the linear operator while at the same time being cheap to apply. No particular rule determines a means of striking this balance; instead, one often relies on heuristics to help form a method for a specific problem. Block preconditioners based on approximating the Schur complement of linear systems have been shown to dramatically improve convergence rates of linear systems related to fluid models [8, 9, 10, 11].

1.1. Outline of Block Preconditioning Methods

Consider the 2×2 block linear system

$$\underbrace{\begin{bmatrix} A & B \\ C & E \end{bmatrix}}_{\mathbb{A}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{\vec{x}} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \end{bmatrix}}_{\vec{b}}. \tag{1}$$

In the context of the shallow water equations, the unknowns correspond to fluid velocity and height. A (right) preconditioned linear system with preconditioning operator $\mathbb{P}$ would be of the form $\mathbb{A}\mathbb{P}^{-1}\mathbb{P}x = b$. $\mathbb{P}$ can be viewed as an approximation of $\mathbb{A}$ based on a formal LDU factorization. If the blocks A and E are invertible, the block LDU factorization is

$$\mathbb{A} = \underbrace{\begin{bmatrix} I & 0 \\ CA^{-1} & I \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} A & 0 \\ 0 & S \end{bmatrix}}_{D} \underbrace{\begin{bmatrix} I & A^{-1}B \\ 0 & I \end{bmatrix}}_{U}, \tag{2}$$

where $S = E - CA^{-1}B$ is the Schur complement of $\mathbb{A}$. Now $\mathbb{A}^{-1}$ can be rewritten as $(LDU)^{-1}$:

$$\underbrace{\begin{bmatrix} I & -A^{-1}B \\ 0 & I \end{bmatrix}}_{U^{-1}} \underbrace{\begin{bmatrix} A^{-1} & 0 \\ 0 & S^{-1} \end{bmatrix}}_{D^{-1}} \underbrace{\begin{bmatrix} I & 0 \\ -CA^{-1} & I \end{bmatrix}}_{L^{-1}}. \tag{3}$$

This formal way of viewing the inverse operation of $\mathbb{A}$ gives insight into the critical operations that are required to solve the linear problem, notably, A^{-1} and S^{-1}. For large problems the factors of the

subsidiary A block cannot be stored, so S cannot be formed. However, this LDU block factorization can serve as a guide to maintain the overall algebraic structure and spectrum of $\mathbb{A}$ and allow physics-based approximations to efficiently capture dominant parts of the subsidiary operators. For spectral-element discretizations, this can be done using domain decomposition methods along the lines developed in [12].

We plan to employ these techniques to atmospheric flow models where subsidiary blocks are composed of a hyperbolic term and an elliptic approximation to the Schur complement. Block preconditioning will allow each operator to be treated separately in order to appropriately exploit their underlying structure and physics. In particular, the hyperbolic operator imposes flow direction that can be used to construct an efficient solver. In contrast, the elliptic nature of the Schur complement has no preferred direction but instead requires information to be propagated throughout the domain rapidly.

1.2. Software Framework

The HOMME-Trilinos framework requires cross-language interoperability between HOMME, written in Fortran, and Trilinos, which is written in C++, using C functions and pointers as mediators. In particular, HOMME uses the Fortran 2003 interface standard which implements the `iso_c_binding` construct. This allows C functions to be called from Fortran and allows Fortran subroutines to be accessed by C. Similarly, the Trilinos/C++ code exports routines to C via the `extern "C"` structure. In conjunction with the `iso_c_binding`, derived data is passed from Fortran to C using the `c_loc` function, which gives the C address of a Fortran pointer. On the C++ side, Fortran derived data is received as type `void *`. Data sent back from C++ to Fortran is recast as a Fortran pointer using the `c_f_pointer` conversion routine. An overview of the general HOMME-Trilinos framework is provided in [6]; extensions of these interfaces have been adopted by other dynamical core components of CESM [13]. Figure 1 uses blue-green gradient boxes to depict where function and data transfers are used in the implicit Homme-Trilinos framework.

A matrix-free, finite-difference Jacobian formulation is a convenient way to introduce implicit timestepping into an explicit code, since the essential parts of F are already supported. Block preconditioners require access to the individual components of the Jacobian or a related linear operator, such as a Picard linearization matrix. We have constructed a preconditioner framework for HOMME-Trilinos that maintains a matrix-free design by implementing the Picard operator as a Fortran subroutine that reuses data structures and code in HOMME for element-based computations. This subroutine is wrapped as an Epetra_Operator via a function pointer so that Trilinos solvers can be used to apply the inverse operation. In Trilinos, linear operators are wrapped in an Epetra_Operator abstraction, which has methods to apply the operator, and (optionally) its inverse, to a vector. With this abstraction for Jacobians and preconditioners, the preconditioned Krylov methods in Trilinos are available, enabling appropriate code reuse in both Trilinos and HOMME.

Figure 1 is a diagram of the nested loop structure required by implicit time integration with a matrix-free, finite-difference Jacobian and a matrix-free block preconditioner P. Blue blocks represent code that is written in Fortran as part of HOMME, and green blocks represent code in C++ as part of Trilinos. From the top level, implicit time integration is performed in Fortran/HOMME; underneath this sits the nonlinear solver which is Newton's method provided by the Trilinos package NOX. Matrix-vector products with the Jacobian, J, centered at a point x, with the vector v can be written as $J_x v = [F(x+\varepsilon v) - F(x)]/\varepsilon$. The evaluation of $F(x)$ is performed outside the linear system solve, whereas the perturbed evaluation $F(x+\varepsilon v)$ occurs as a part of each matrix-vector product inside the iterative linear solver. FGMRES [14] is used to solve the the linear Jacobian system, since the preconditioner can differ slightly during each linear iteration. FGMRES is provided by the Trilinos package Belos. The matrix-free Jacobian and preconditioner are set by using the `NOX::Epetra::LinearSystemStratimikos` class. We have implemented the matrix-free block preconditioner through a class that wraps a function pointer to the Fortran routine for the forward operator P as an Epetra_Operator. This Epetra_Operator is then used as the operator that the Belos package uses to evaluate the action of P^{-1} via GMRES. Future development will allow for block seggregation of P in order to use packages that support multilevel

evaluation of the individual blocks instead of GMRES.

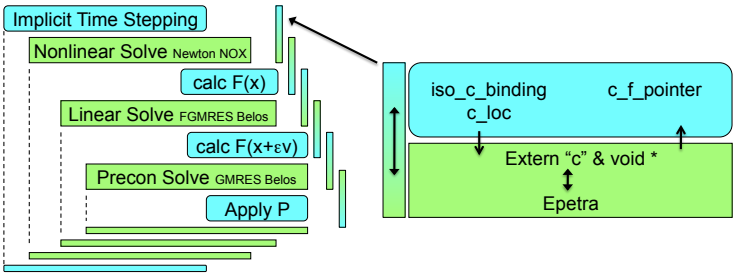

Figure 1. Program flow within the Homme-Trilinos framework. Blue blocks are performed within HOMME; green blocks are performed using Trilinos. Gradient colored boxes denote function and data transfers between codes. C extensions to Fortran and C++ allow for interoperability.

1.3. Conclusions and Future Work

The software infrastructure to support implicit time-stepping with block preconditioners in HOMME is in place. Employing a completely matrix-free preconditioning framework using multiple Trilinos packages requires careful attention to data transfers between Trilinos and HOMME. Verification of the preconditioning framework is under way using parallel tests on the Cray XT5 machines at ORNL (jaguarpf). Future work includes investigating efficiency of the block preconditioning approach on problems using time steps well above the gravity wave speed.

2. Superfast linear solvers and preconditioners

We have designed a parallel structured multifrontal algorithm that maintains the nearly linear complexity as the sequential one, while at the same time minimizing the amount of communication as much as possible. Below is the outline of our parallel algorithm.

(i) Use a parallel nested dissection algorithm (e.g., ParMetis or PT-Scotch) to reorder the matrix and obtain the assembly tree with separator nodes.

(ii) Perform parallel numerical factorization in two phases with P processors: (1) Lower-level phase: perform classical multifrontal factorization. This is first done in serial at the bottom-most levels with P subtrees, then in parallel at the higher levels when there are fewer than P separators; (2) Upper-level phase: perform parallel *structured* multifrontal factorization.

- At the switching level, construct a *Hierarchically SemiSeparable (HSS)* representation for the frontal matrices in parallel.
- At each higher level, first eliminate a separator using a parallel HSS factorization algorithm and then perform parallel extend-add to merge the update matrices to the parent.

The success of this approach depends crucially on an efficient handling of the hierarchical parallelism: the outer coarse-grained parallelism related to the multifrontal factorization, which is guided by the assembly tree, and the inner fine-grained parallelism requiring various HSS operations. The outer parallel algorithm has been established by previous research, most notably implemented in the widely used MUMPS software package [15]. Our novel work lies in the development of the parallel HSS matrix kernels with low communication complexity, and seamlessly integrating them into the outer parallel structure [16]. These kernels include parallel HSS compression via rank revealing QR (RRQR) factorization for frontal matrix construction, parallel HSS QL/LQ (i.e., ULV) factorization for frontal matrix elimination, and parallel data-sparse/structured extend-add for assembling update matrices.

Based on the above parallel HSS algorithms, we have implemented an HSS-embedded multifronal solver for the standard stencils on regular Cartesian meshes. We obtained significant speedup and memory saving over the classical multifrontal solver when applied to the 2D anisotropic Helmholtz equations for seismic imaging. We were able to solve a linear system with 6.4 billion unknowns using 4,096 processors of the Cray XE6 machines at NERSC (hopper), in less than 20 minutes. The classical

multifrontal factorization simply failed because of high demand of memory. For smaller problems for which both methods worked, the HSS-embedded solver is usually 2–3x faster [17].

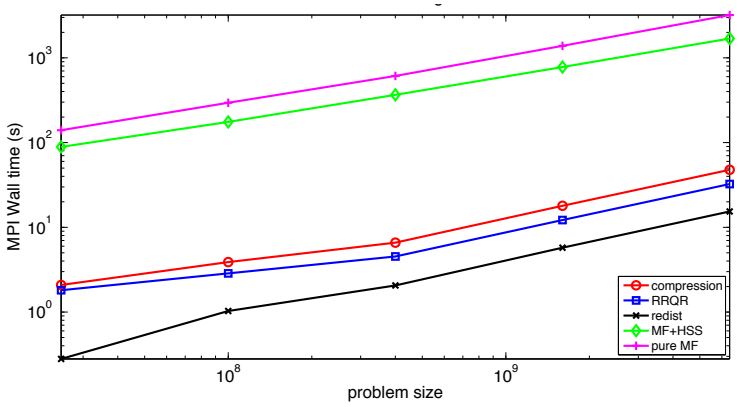

Figure 2. Weak-scaling test of the new superfast linear solver on the Cray XE6 (hopper at NERSC). 2D mesh $N \times N$: 5000, 10000, 20000, 40000, 80000, processor counts: 16, 64, 256, 1024, 4096. Up to 6.4 billion unknowns.

Acknowledgments

Portions of this research used resources of the National Center for Computational Sciences at Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract DE-AC05-00OR22725. The work was supported in part by the Director, Office of Science, Office of Advanced Scientific Computing Research of the U.S. Department of Energy under Contract No. D-AC02-05CH11231. This work performed in part under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

References

[1] Washington W M, Buja L and Craig A 2009 *Philosophical Transactions of the Royal Society A* **367** 833–846
[2] Taylor M A, Tribbia J and Iskandarani M 1997 *Journal of Computational Physics* **130** 92–108
[3] Courant R, Friedrichs K and Lewy H 1967 *IBM Journal* 215–234
[4] Keyes D, Reynolds D and Woodward C 2006 *Journal of Physics: Conference Series* **46** 433–442
[5] Heroux M, Bartlett R, Hoekstra V H R, Hu J, Kolda T, Lehoucq R, Long K, Pawlowski R, Phipps E, Salinger A, Thornquist H, Tuminaro R, Willenbring J and Williams A 2003 An Overview of Trilinos Tech. Rep. SAND2003-2927 Sandia National Laboratories
[6] Evans K J, Rouson D W I, Salinger A G, Taylor M A, Weijer W and III J B W 2009 *A Scalable and Adaptable Solution Framework within Components of the Community Climate System Model* (*Lecture Notes in Computer Science* vol 5545) (Springer) pp 332–341
[7] Evans K J, Taylor M A and Drake J B 2010 *Monthly Weather Review* **138** 3333–3341
[8] Elman H C, Howle V E, Shadid J, Shuttleworth R and Tuminaro R 2008 *Journal of Computational Physics* **227** 1790–1808
[9] Elman H C, Howle V E, Shadid J, Silvester D and Tuminaro R 2007 *SIAM Journal on Scientific Computing* **30** 290–311
[10] Elman H C, Silvester D and Wathen A 2005 *Finite Elements and Fast Iterative Solvers with Applications in Incompressible Fluid Dynamics* Numerical Mathematics and Scientific Computation (New York: Oxford University Press)
[11] Lott P A and Elman H C In preparation Fast solvers for incompressible Navier-Stokes equations with spectral elements
[12] Lott P A and Elman H C 2011 *Numerical Methods in Partial Differential Equations* **27** 231–254
[13] Evans K J, Salinger A G, Worley P H, Price S, Lipscomb W, Nichols J, III J B W, Perego M, Edwards J, Vertenstein M and Lemieux J F submitted *IEEE Software*
[14] Saad Y 1993 *SIAM Journal on Scientific Computing* **14** 461–469
[15] MUMPS: A MUltifrontal Massively Parallel sparse direct Solver `http://graal.ens-lyon.fr/MUMPS/`
[16] Wang S, Li X, Xia J, Situ Y and de Hoop M V 2011 *SIAM J. Scientific Computing* (submitted)
[17] Wang S, Xia J, de Hoop M V and Li X 2011 *Geophysics* (sumitted)